

DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUU	UUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUU	UUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUU	UUU	GGGGGGGGGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDD	DDD	BBB	UUU	UUU	GGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	GGGGGGGGGG

```
DDDDDDDD  BBBB888  GGGGGGGG  TTTTTTTTTT  BBBB8888  KK      KK
DDDDDDDD  BBBB8888  GGGGGGGG  TTTTTTTTTT  BBBB8888  KK      KK
DD      DD  BB      BB  GG      GG      TT      TT      BB      BB  KK      KK
DD      DD  BB      BB  GG      GG      TT      TT      BB      BB  KK      KK
DD      DD  BB      BB  GG      GG      TT      TT      BB      BB  KK      KK
DD      DD  BB      BB  GG      GG      TT      TT      BB      BB  KK      KK
DD      DD  BBBB8888  GG      GG      TT      TT      BBBB8888  KKKKKK  KK
DD      DD  BBBB8888  GG      GG      TT      TT      BBBB8888  KKKKKK  KK
DD      DD  BB      BB  GG      GG      TT      TT      BB      BB  KK      KK
DD      DD  BB      BB  GG      GG      TT      TT      BB      BB  KK      KK
DD      DD  BB      BB  GG      GG      TT      TT      BB      BB  KK      KK
DD      DD  BB      BB  GG      GG      TT      TT      BB      BB  KK      KK
DDDDDDDD  BBBB8888  GGGGGG      TT      TT      BBBB8888  KK      KK
DDDDDDDD  BBBB8888  GGGGGG      TT      TT      BBBB8888  KK      KK
                                     ....
                                     ....
                                     ....
                                     ....
```

```
LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
```

```
1 0001 0 MODULE DBGTBK ( IDENT = 'V04-000' ) =
2 0002 1 BEGIN
3 0003 1
4 0004 1 *****
5 0005 1 *
6 0006 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
7 0007 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
8 0008 1 * ALL RIGHTS RESERVED.
9 0009 1 *
10 0010 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
11 0011 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
12 0012 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
13 0013 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
14 0014 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
15 0015 1 * TRANSFERRED.
16 0016 1 *
17 0017 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
18 0018 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
19 0019 1 * CORPORATION.
20 0020 1 *
21 0021 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
22 0022 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
23 0023 1 *
24 0024 1 *****
25 0025 1
26 0026 1
27 0027 1 MODULE FUNCTION
28 0028 1 This module contains the routines that implement the SHOW CALLS
29 0029 1 command. These routines give a traceback from the program location
30 0030 1 where the user is currently stopped.
31 0031 1
32 0032 1 AUTHOR: Carol Peters, CREATION DATE: September 20, 1977
33 0033 1
34 0034 1 MODIFIED BY:
35 0035 1 Mike Candela, 28 January 1980
36 0036 1 V. Holt, 14 May 1982
37 0037 1
38 0038 1 1.01 25-SEP-78 MCC Deleted require file SYSLIT
39 0039 1 1.02 9-OCT-78 MCC Traceback reporting corrected to terminate when
40 0040 1 current FP = addr of DBG$FINAL_HANDL (bug-fix)
41 0041 1 1.03 02-NOV-78 DAR Removed check for FORTRAN MODULE from dbg$traceback.
42 0042 1 Also put in explicit field lengths into FAO strings.
43 0043 1 1.04 03-NOV-78 DAR Traceback also stops if the PC is DBG$PSEUDO_PROG
44 0044 1 1.05 30-NOV-79 JBD Put in statement number support.
45 0045 1 1.06 28-JAN-80 MCC Fixed out_traceback to correctly format 31
46 0046 1 routine and module names for SHOW CALLS
47 0047 1
48 0048 1 1.07 19-apr-80 ala Added additional parameter to output routines
49 0049 1 to allow access to output buffer's address
50 0050 1 3B.0 01-Mar-82 PS When dbg$val_to_sym corresponds pc to rstptr,
51 0051 1 check to see if this is a data symbol before
52 0052 1 search for the surrounding routine and module
53 0053 1 entries.
54 0054 1 3B.0 27-Apr-82 PS Display the module name when SHOW CALL even if
55 0055 1 the module is not set.
56 0056 1 3B.1 14-May-82 VJH Added call to DBG$FLUSHBUF, eliminating need to
57 0057 1 initialize local output buffer.
```


58	0058	1	3B.2	3-Jun-82	VJM	Removed all references to DBG\$FAO_PUT and
59	0059	1				DBG\$OUT_PUT, as these are now obsolete.
60	0060	1				Replaced them with calls to DBG\$PRINT and
61	0061	1				DBG\$NEWLINE, respectively.
62	0062	1	3B.2	16-Nov-82	PS	Do a gernal clean up. (We always print module
63	0063	1				name from the SAT look up for the current pc.
64	0064	1				We mark the set module. We print JSB message.
65	0065	1				We print EXC message.)
66	0066	1	3B.2	27-Dec-82	BB	Clean up style and other minor things.
67	0067	1				
68	0068	1				
69	0069	1				
70	0203	1				REQUIRE 'SRC\$:DBGPROLOG.REQ';
71	0204	1				LIBRARY 'LIB\$:DBGGEN.L32';
72	0205	1				
73	0206	1				FORWARD ROUTINE
74	0207	1				DBG\$TRACEBACK: NOVALUE,
75	0208	1				
76	0209	1				FIND_MODRST,
77	0210	1				
78	0211	1				OUT_TRACEBACK: NOVALUE;
79	0212	1				

```

: Traces calls through the stack and
:   generates the SHOW CALLS output
: Find the module RST pointer for a PC
:   from the Program SAT
: Output a single line of traceback
:   information

```

```

: 81      0213 1 EXTERNAL ROUTINE
: 82      0214 1     DBG$FINAL_HANDL,
: 83      0215 1     DBG$PC_TO_LINE_LOOKUP,
: 84      0216 1     DBG$PRINT : NOVALUE,
: 85      0217 1     DBG$NEWLINE : NOVALUE,
: 86      0218 1     DBG$SEARCH_BIN SAT,
: 87      0219 1     DBG$STA_SYMNAME: NOVALUE,
: 88      0220 1     DBG$PC_TO_SYMID,
: 89      0221 1     SYS$GETMSG;
: 90      0222 1
: 91      0223 1 EXTERNAL
: 92      0224 1     DBG$PSEUDO_EXIT,
: 93      0225 1     DBG$RUNFRAME: BLOCK[,BYTE],
: 94      0226 1     SAT$START_ADDR;

```

```

: Call frame exception handler
: Translates a PC to a line number
: Format output lines.
: Flush output lines.
: Search-SAT routine
: Get symbol's name
: Translates a value to an RST pointer.
: Get the message text for a condition

: Point to which CALL returns
: The current register runframe
: Starting address of Program SAT

```

```

96 0227 1 GLOBAL ROUTINE DBG$TRACEBACK(INITIAL_PC, FP_POINTER,
97 0228 1 EXCEPTION_NAME, NUM_LEVELS): NOVALUE =
98 0229 1
99 0230 1 FUNCTION
100 0231 1 This routine collects the symbolic information describing each
101 0232 1 stack frame starting at the stack frame pointed to by the user's
102 0233 1 FP, and proceeding through the frame with which the user program
103 0234 1 was called by CLI, by the OTS, or by DEBUG.
104 0235 1
105 0236 1 Once the symbolic information for a frame is collected, a routine
106 0237 1 is called to output this information to DBG$OUTPUT.
107 0238 1
108 0239 1 The num_levels parameter is either -1, or it is the
109 0240 1 number of call frames which the user has specifically
110 0241 1 requested (via SHOW CALLS N).
111 0242 1
112 0243 1 INPUTS
113 0244 1 INITIAL_PC - PC of user program when traceback occurs
114 0245 1
115 0246 1 FP_POINTER - FP of user program when traceback occurs
116 0247 1
117 0248 1 EXCEPTION_NAME - Type of exception where:
118 0249 1 1 - trap type exception
119 0250 1 2 - fault or abort type exception
120 0251 1
121 0252 1 NUM_LEVELS - The number of frames the user wants to see,
122 0253 1 or -1 which implies "show them all".
123 0254 1
124 0255 1
125 0256 1 OUTPUTS
126 0257 1 NONE
127 0258 1
128 0259 1
129 0260 2 BEGIN
130 0261 2
131 0262 2 BUILTIN
132 0263 2 PROBER; ! Probe for read access to a location
133 0264 2
134 0265 2 LITERAL
135 0266 2 MAX_STRING_SIZE = 256; ! ???
136 0267 2
137 0268 2 LOCAL
138 0269 2 CALL_FLAG, ! Flag to indicate the call is from
139 0270 2 ! JSB or BSB
140 0271 2 CURRENT_FP : REF BLOCK[BYTE], ! Value of FP of working stack frame
141 0272 2 CURRENT_PC, ! Current PC in writable variable
142 0273 2 EXC_TYPE, ! Type of exception
143 0274 2 J ! Index value used for several purposes
144 0275 2 LINE_NUMBER, ! Matching line number
145 0276 2 MODNAME, ! Pointer to module's name
146 0277 2 MOD_RSTPTR: REF RST$ENTRY, ! Pointer to RST entry for outermost scope
147 0278 2 MOD_SET_FLAG, ! Flag to indicate that module is SET
148 0279 2 MSG_DESCR: DBG$STG_DESC, ! String descriptor for message text
149 0280 2 MSGLEN: WORD, ! The length of the message text
150 0281 2 MSG_STRING: ! The message text buffer
151 0282 2 VECTOR[MAX_STRING_SIZE, BYTE], !
152 0283 2 NEXT_FP: REF BLOCK[BYTE], ! ???
```



```
153 0284 2 REGMASK: BITVECTOR[16], ! The register save mask bit vector
154 0285 2 REGSAVELOC: REF VECTOR[.LONG], ! Pointer to the register save area in
155 0286 2 ! the current call frame
156 0287 2 RTN_RSTPTR: REF RST$ENTRY, ! Pointer to RST entry for routine
157 0288 2 SAVED_RUNFRAME: REF BLOCK[.BYTE], ! Pointer to saved runframe from the
158 0289 2 ! DEBUG CALL command
159 0290 2 SIG_VECTOR: REF VECTOR[.LONG], ! Pointer to the Signal Argument Vector
160 0291 2 SPVALUE: REF VECTOR[.LONG], ! The value of SP in the current frame
161 0292 2 SYM_DSTPTR: REF DST$RECORD, ! Pointer to corresponding DST entry
162 0293 2 SYMNAME, ! Pointer to symbol's name
163 0294 2 SYM_RSTPTR: REF RST$ENTRY, ! Pointer to RST entry from VAL_TO_SYM
164 0295 2 STARTING_PC, ! PC of start of routine or module
165 0296 2 START_PC, ! ???
166 0297 2 END_PC, ! ???
167 0298 2 STMT_NUMBER; ! Matching statement number
168 0299
169 0300
170 0301
171 0302 ! If the user doesn't want to see any frames just return. Otherwise check
172 0303 ! that some call frames are active, get values of PC and FP to use, and
173 0304 ! set up the exception type.
174 0305
175 0306 IF .NUM_LEVELS EQL 0 THEN RETURN;
176 0307 IF .INITIAL_PC EQL 0 THEN SIGNAL(DBG$_NOCALLS);
177 0308
178 0309
179 0310 ! Initialization.
180 0311
181 0312 NEXT_FP = .FP_POINTER;
182 0313 CURRENT_PC = .INITIAL_PC;
183 0314 EXC_TYPE = .EXCEPTION_NAME;
184 0315 CALC_FLAG = FALSE;
185 0316 SAVED_RUNFRAME = .DBG$RUNFRAME[DBG$$_NEXT_LINK];
186 0317
187 0318
188 0319 ! Print the SHOW CALLS header.
189 0320
190 0321 DBG$PRINT(UPLIT BYTE (%ASCIC
191 0322 ' module name routine name line rel PC abs PC!/'));
192 0323 DBG$NEWLINE();
193 0324
194 0325
195 0326 ! The following loop translates the current PC into a routine name and then
196 0327 ! prints the name of the surrounding module, the name of the routine, the
197 0328 ! line number, and the relative and absolute PC values for each user stack
198 0329 ! frame.
199 0330
200 0331 INCR DEPTH FROM 0 TO MINU(.NUM_LEVELS, 1000) - 1 DO
201 0332 BEGIN
202 0333 IF PROBER(%REF(0), %REF(20), NEXT_FP[SF$A_HANDLER]) EQL 0
203 0334 THEN
204 0335 BEGIN
205 0336 SIGNAL(DBG$_BADSTACK);
206 0337 RETURN;
207 0338 END;
208 0339
209 0340
```

```
210 0341 4
211 0342 4
212 0343 4
213 0344 4
214 0345 4
215 0346 4
216 0347 4
217 0348 4
218 0349 4
219 0350 4
220 0351 4
221 0352 4
222 0353 4
223 0354 4
224 0355 4
225 0356 4
226 0357 4
227 0358 4
228 0359 4
229 0360 4
230 0361 3
231 0362 4
232 0363 4
233 0364 4
234 0365 4
235 0366 4
236 0367 4
237 0368 4
238 0369 4
239 0370 4
240 0371 5
241 0372 5
242 0373 4
243 0374 4
244 0375 4
245 0376 4
246 0377 4
247 0378 4
248 0379 4
249 0380 4
250 0381 4
251 0382 4
252 0383 4
253 0384 4
254 0385 4
255 0386 4
256 0387 4
257 0388 4
258 0389 4
259 0390 4
260 0391 4
261 0392 4
262 0393 4
263 0394 4
264 0395 4
265 0396 4
266 0397 4

! Stop if the exception handler address points to DEBUG's final handler.
! This indicates that we have reached the end of the call stack.
IF .NEXT_FP[SF$A_HANDLER] EQL DBG$FINAL_HANDL THEN RETURN;

! Abort the SHOW CALLS processing if the user entered Control-Y DEBUG
! to stop the current command.
$ABORT_ON_CONTROL_Y;

! Check to see if this is an exception handler. (A handler is recog-
! nized by having a return PC of hex 80000014, which is where the VMS
! exception handling mechanism calls user handlers.) If it is an
! exception handler, we must get the exception PC from the signal
! argument list. The location of this list is computed from the stack
! pointer value.
IF (.CURRENT_PC EQL XX'80000014') AND (.DEPTH NEQ 0)
THEN
  BEGIN

    ! Pass the saved registers in this call frame.
    REGMASK = .CURRENT_FP[SF$W_SAVE_MASK];
    REGSAVELOC = CURRENT_FP[SF$L_SAVE_REGS];
    J = 0;
    INCR J FROM 0 TO 11 DO
      BEGIN
        IF .REGMASK[J] THEN J = .J + 1;
      END;

    ! Set the stack pointer points at the end of the saved registers.
    ! Adjust it by the offset values.
    SPVALUE = REGSAVELOC[J];
    SPVALUE = .SPVALUE + .CURRENT_FP[SF$V_STACKOFFS];

    ! Pass one longword of junk and the argument count.
    SPVALUE = .SPVALUE + 8;

    ! Get the pointer to the signal argument list. Pick up the PC of
    ! the signal from the signal argument list. Then print the line
    ! identifying this routine as a condition handler and the line that
    ! displays the message text that identifies the signalled condition.
    SIG_VECTOR = .SPVALUE[0];
    J = .SIG_VECTOR[0];
    CURRENT_PC = .SIG_VECTOR[J - 1];
    DBG$PRINT(UPLOT BYTE($ASCII
      '----- above condition handler called with exception !XL'),
```



```
267      SIG VECTOR[1];
268      DBG$PRINT(UPLOT BYTE(%ASCIC ':'));
269      DBG$NEWLINE();
270      MSG_DESCR[DSC$W_LENGTH] = MAX_STRING_SIZE;
271      MSG_DESCR[DSC$A_POINTER] = MSG_STRING;
272      SYS$GETMSG(.SIG_VECTOR[1], MSGLEN, MSG_DESCR, 0, 0);
273      MSG_DESCR[DSC$W_LENGTH] = MSGLEN;
274      DBG$PRINT(UPLOT BYTE(%ASCIC '----- !AS'), MSG_DESCR);
275      DBG$NEWLINE();
276      END;
277
278      ! Check to see if the CURRENT_PC is caused by the DEBUG CALL command.
279      ! If so, print the line that indicates this and pick up the actual
280      ! user PC value from the saved run-frame for this CALL command.
281      IF .CURRENT_PC EQL DBG$PSEUDO_EXIT
282      THEN
283      BEGIN
284      CURRENT_PC = .SAVED_RUNFRAME[DBG$L_USER_PC];
285      DBG$PRINT(UPLOT BYTE(%ASCIC
286      '----- above routine called from DEBUG CALL command'));
287      DBG$NEWLINE();
288      SAVED_RUNFRAME = .SAVED_RUNFRAME[DBG$L_NEXT_LINK];
289      EXC_TYPE = FAULT_EXC;
290      END;
291
292      ! Obtain the name of the innermost routine that surrounds the address.
293      ! If there is no such routine in the RST, find out what module it is
294      ! in and print only the module name (if any) and the absolute PC value.
295      IF NOT DBG$PC_TO_SYMD(.CURRENT_PC, SYM_RSTPTR, TRUE)
296      THEN
297      BEGIN
298      MODNAME = 0;
299      MOD_SET_FLAG = FALSE;
300      MOD_RSTPTR = FIND_MODRST(.CURRENT_PC);
301      IF .MOD_RSTPTR NEQ 0
302      THEN
303      BEGIN
304      DBG$STA_SYMD(.MOD_RSTPTR, MODNAME);
305      MOD_SET_FLAG = .MOD_RSTPTR[RST$V_MODSET];
306      END;
307      OUT_TRACEBACK (.MODNAME, 0, 0, 0, 0, .CURRENT_PC, .MOD_SET_FLAG);
308      END
309      ELSE
310      BEGIN
311      IF .SYM_RSTPTR EQL 0 THEN $DBG_ERROR('DBGTBK\TRACEBACK');
312      SYM_DSTPTR = .SYM_RSTPTR[RST$L_DSTPTR];
313      IF .SYM_RSTPTR[RST$V_GLOBAL]
314      THEN
315      ! Routine found in GST rather than in RST. (This is the case if
```

the module containing the routine is not set). Just print the routine name and the relative and absolute PC values.
Note: Now the routine will find the module RST pointer thru Program SAT, and print out the module name even if the module is not set.

```
BEGIN
CALL FLAG = TRUE;
MOD_RSTPTR = FIND_MODRST(.CURRENT_PC);
DBG$STA_SYMNAME(.SYM_RSTPTR, SYMNAME);
IF .MOD_RSTPTR NEQ 0
THEN
  BEGIN
    DBG$STA_SYMNAME(.MOD_RSTPTR, MODNAME);
    OUT_TRACEBACK (.MODNAME, SYMNAME, 0, 0,
      (.CURRENT_PC - .SYM_DSTPTR[.DST$L_VALUE]),
      .CURRENT_PC, .MOD_RSTPTR[RST$V_MODSET]);
  END
ELSE
  OUT_TRACEBACK (0, .SYMNAME, 0, 0,
    (.CURRENT_PC - .SYM_DSTPTR[.DST$L_VALUE]),
    .CURRENT_PC);
END
ELSE
  BEGIN
    IF .SYM_RSTPTR[RST$B_KIND] EQL RST$K_DATA
    THEN
      OUT_TRACEBACK(0, 0, 0, 0, 0, .CURRENT_PC)
    ELSE
      BEGIN
        ! Search for the surrounding routine and module entries.
        !
        CALL FLAG = TRUE;
        RTN_RSTPTR = 0;
        MOD_RSTPTR = .SYM_RSTPTR;
        WHILE .MOD_RSTPTR NEQ 0 DO
          BEGIN
            CASE .MOD_RSTPTR[RST$B_KIND] FROM RST$K_TYPE_MINIMUM
              TO RST$K_TYPE_MAXIMUM OF
              SET
                [RST$K_MODULE]:
                  EXITLOOP;
                [RST$K_ROUTINE]:
                  IF .RTN_RSTPTR EQL 0
                  THEN
                    BEGIN
                      SYM_RSTPTR = RTN_RSTPTR = .MOD_RSTPTR;
                      SYM_DSTPTR = .MOD_RSTPTR[RST$L_DSTPTR];
                    END;
              END;
          END;
        END;
      END;
  END;
END;
```

381 0512 7
382 0513 7
383 0514 7
384 0515 7
385 0516 7
386 0517 7
387 0518 7
388 0519 7
389 0520 7
390 0521 7
391 0522 7
392 0523 7
393 0524 7
394 0525 7
395 0526 6
396 0527 6
397 0528 6
398 0529 6
399 0530 6
400 0531 6
401 0532 6
402 0533 6
403 0534 6
404 0535 7
405 0536 7
406 0537 7
407 0538 6
408 0539 6
409 0540 6
410 0541 6
411 0542 6
412 0543 6
413 0544 6
414 0545 6
415 0546 6
416 0547 6
417 0548 7
418 0549 7
419 0550 7
420 0551 7
421 0552 7
422 0553 7
423 0554 7
424 0555 7
425 0556 6
426 0557 6
427 0558 6
428 0559 6
429 0560 6
430 0561 6
431 0562 3
432 0563 3
433 0564 4
434 0565 4
435 0566 3
436 0567 3
437 0568 3

```

[ RST$K_ENTRY,
  RST$K_BLOCK,
  RST$K_LINE,
  RST$K_LABEL ]:
0;

[ INRANGE, OUTRANGE ] :
  SIGNAL( DBGS_RSTERR );

TES;

MOD_RSTPTR = .MOD_RSTPTR[ RST$K_UPSCOPEPTR ];
IF .MOD_RSTPTR EQ 0 THEN SIGNAL( DBGS_RSTERR );

END;          ! End of WHILE loop

RTN_RSTPTR = .SYM_RSTPTR;
STARTING_PC = .SYM_DSTPTR[ DST$K_VALUE ];
IF NOT DBGS_PC TO LINE_LOOKUP
  ( .CURRENT_PC - ( .EXC_TYPE neq FAULT_EXC ),
    LINE_NUMBER, STMT_NUMBER,
    START_PC, END_PC, .MOD_RSTPTR )
THEN
  BEGIN
    LINE_NUMBER = 0;
    STMT_NUMBER = 0;
  END;

! We always use the MODRST ptr from searching module and
! Program Static Address Table for the given current PC.
MOD_RSTPTR = FIND_MODRST( .CURRENT_PC );
DBGS_STA_SYMNAME( .SYM_RSTPTR, SYMNAME );
IF .MOD_RSTPTR NEQ 0
THEN
  BEGIN
    DBGS_STA_SYMNAME( .MOD_RSTPTR, MODNAME );
    OUT_TRACEBACK ( .MODNAME, .SYMNAME,
      .LINE_NUMBER, .STMT_NUMBER,
      .CURRENT_PC - .STARTING_PC,
      .CURRENT_PC, .MOD_RSTPTR[ RST$K_MODSET ] );
  END
ELSE
  OUT_TRACEBACK ( 0, .SYMNAME,
    .LINE_NUMBER, .STMT_NUMBER,
    .CURRENT_PC - .STARTING_PC,
    .CURRENT_PC );

END;          ! End of Searching for routine and modules.

END;          ! End of Checking data symbol rstptr.

END;

```



```

438 0569 IF .CALL_FLAG
439 0570 THEN
440 0571 BEGIN
441 0572 CALL_FLAG = FALSE;
442 0573 IF .SYM_RSTPTR[RST$B_KIND] EQL RST$K_ROUTINE
443 0574 THEN
444 0575 BEGIN
445 0576 IF (.CURRENT_PC GEQU .SYM_RSTPTR[RST$L_STARTADDR]) AND
446 0577 (.CURRENT_PC LEQU .SYM_RSTPTR[RST$L_ENDADDR])
447 0578 THEN
448 0579 BEGIN
449 0580 SYM_DSTPTR = .SYM_RSTPTR[RST$L_DSTPTR];
450 0581 IF .SYM_DSTPTR[DST$V_RTNBEG_NO_CALL]
451 0582 THEN
452 0583 BEGIN
453 0584 DBG$PRINT(UPLOT BYTE(%ASCII
454 0585 '----- above JSB routine called from unknown location'));
455 0586 DBG$NEWLINE();
456 0587 END;
457 0588
458 0589 END;
459 0590
460 0591 END;
461 0592
462 0593 END;
463 0594
464 0595
465 0596 ! Update CURRENT_PC and CURRENT_FP to the previous frame. Set the
466 0597 ! FP to point to next frame stack.
467 0598
468 0599 EXC_TYPE = TRAP_EXC;
469 0600 CURRENT_FP = .NEXT_FP;
470 0601 CURRENT_PC = .NEXT_FP[SF$L_SAVE_PC];
471 0602 NEXT_FP = .NEXT_FP[SF$L_SAVE_FP];
472 0603 END; ! End of DECR loop through call stack
473 0604
474 0605
475 0606 ! We have output as many traceback lines as the user requested. Now return.
476 0607
477 0608 RETURN;
478 0609
479 0610 END;

```

INFO#250 L1:0367
Referenced LOCAL symbol CURRENT_FP is probably not initialized

														.TITLE	DBGTBK						
														.IDENT	\V04-000\						
														.PSECT	DBG\$PLIT,NOWRT,	SHR,	PIC,0				
20	20	65	6D	61	6E	20	65	6C	75	64	6F	6D	20	4F	00000	P.AAA:	.ASCII	\0 module name	routine name	\	
65	6D	61	6E	20	65	6E	69	74	75	6F	72	20	20	20	0000F						
					20	20	20	20	20	20	20	20	20	20	0001E						
65	6E	69	6C	20	20	20	20	20	20	20	20	20	20	20	00028		.ASCII	\	line	rel PC	abs PC!/\
20	20	43	50	20	6C	65	72	20	20	20	20	20	20	20	00037						
					2F	21	43	50	20	73	62	61	20	20	00046						

[illegible]

			50	D5	0006B	48:	TSTL	R0			
			0E	12	0006D		BNEQ	58			
		00028F28	8F	DD	0006F		PUSHL	#167720		0336	
			01	FB	00075		CALLS	#1, LIB\$SIGNAL			
				04	0007C		RET			0335	
		50 00000000G	00	9E	0007D	58:	MOVAB	DBG\$FINAL_HANDL, R0		0344	
		50	68	D1	00084		CMPL	(NEXT_FP), R0			
			01	12	00087	68:	BNEQ	78			
				04	00089		RET				
0D	00000000G	00	01	E1	0008A	78:	BBC	#1, DBG\$GV_CONTROL+1, 88			
		000280E8	8F	DD	00092		PUSHL	#164072			
	00000000G	00	01	FB	00098		CALLS	#1, LIB\$SIGNAL			
	80000014	8F	53	D1	0009F	88:	CMPL	CURRENT_PC, #-2147483628		0360	
			03	13	000A6		BEQL	108			
			009B	31	000AB	98:	BRW	138			
			04	AE	D5	000AB	108:	TSTL	DEPTH		
				F8	13	000AE		BEQL	98		
	20	AE	06	A9	B0	000B0		MOVW	6(CURRENT_FP), REGMASK	0367	
	1C	AE	14	A9	9E	000B5		MOVAB	20(R9), REGSAVELOC	0368	
				56	D4	000BA		CLRL	J	0369	
				50	D4	000BC		CLRL	I	0370	
02	20	AE		50	E1	000BE	118:	BBC	I, REGMASK, 128	0372	
				56	D6	000C3		INCL	J		
F5		50		0B	F3	000C5	128:	AOBLEQ	#11, I, 118	0370	
			1C	BE	46	DE	000C9	MOVAL	@REGSAVELOC[J], SPVALUE	0379	
50	07	A9		06	EF	000CF		EXTZV	#6, #2, 7(CURRENT_FP), R0	0380	
				50	C0	000D5		ADDL2	R0, SPVALUE		
				08	C0	000D9		ADDL2	#8, SPVALUE	0385	
				57	BE	D0	000DD	MOVL	@SPVALUE, SIG_VECTOR	0393	
				56	D0	000E1		MOVL	(SIG_VECTOR), J	0394	
				53	FC	A746	D0	000E4	MOVL	-4(SIG_VECTOR)[J], CURRENT_PC	0395
				04	A7	DD	000E9	PUSHL	4(SIG_VECTOR)	0398	
					EF	9F	000EC	PUSHAB	P.AAB	0396	
	00000000G	00		02	FB	000F2		CALLS	#2, DBG\$PRINT		
					EF	9F	000F9	PUSHAB	P.AAC	0399	
	00000000G	00		01	FB	000FF		CALLS	#1, DBG\$PRINT		
	00000000G	00		00	FB	00106		CALLS	#0, DBG\$NEWLINE	0400	
	F4	AD	0100	8F	B0	0010D		MOVW	#256, MSG_DESCR	0401	
	F8	AD	48	AE	9E	00113		MOVAB	MSG_STRING, MSG_DESCR+4	0402	
				7E	7C	00118		CLRW	-(SP)	0403	
				F4	AD	9F	0011A	PUSHAB	MSG_DESCR		
				30	AE	9F	0011D	PUSHAB	MSG[EN		
				04	A7	DD	00120	PUSHL	4(SIG_VECTOR)		
	00000000G	00		05	FB	00123		CALLS	#5, SYS\$GETMSG		
	F4	AD	24	AE	B0	0012A		MOVW	MSGLEN, MSG_DESCR	0404	
				F4	AD	9F	0012F	PUSHAB	MSG_DESCR	0405	
					EF	9F	00132	PUSHAB	P.AAD		
	00000000G	00		02	FB	00138		CALLS	#2, DBG\$PRINT		
	00000000G	00		00	FB	0013F		CALLS	#0, DBG\$NEWLINE	0406	
		50	00000000G	00	9E	00146	138:	MOVAB	DBG\$PSEUDO_EXIT, R0	0414	
		50		53	D1	0014D		CMPL	CURRENT_PC, R0		
				1F	12	00150		BNEQ	148		
		53	40	AA	D0	00152		MOVL	64(SAVED_RUNFRAME), CURRENT_PC	0417	
				EF	9F	00156		PUSHAB	P.AAE	0418	
	00000000G	00		01	FB	0015C		CALLS	#1, DBG\$PRINT		
	00000000G	00		00	FB	00163		CALLS	#0, DBG\$NEWLINE	0420	
		5A		6A	D0	0016A		MOVL	(SAVED_RUNFRAME), SAVED_RUNFRAME	0421	

		10	AE		02	DD	0016D	MOVL	#2, EXC_TYPE	0422	
					01	DD	00171	PUSHL	#1	0430	
				2C	AE	9F	00173	PUSHAB	SYM_RSTPTR		
	00000000G		00		53	DD	00176	PUSHL	CURRENT_PC		
			36		03	FB	00178	CALLS	#3, DBG\$PC_TO_SYMD		
					50	E8	0017F	BLBS	R0, 16\$		
				44	AE	D4	00182	CLRL	MODNAME	0433	
				14	AE	D4	00185	CLRL	MOD_SET_FLAG	0434	
					53	DD	00188	PUSHL	CURRENT_PC	0435	
	0000V		CF		01	FB	0018A	CALLS	#1, FIND_MODRST		
	2C		AE		50	DD	0018F	MOVL	R0, MOD_RSTPTR		
			52		2C	AE	DD	00193	MOD_RSTPTR, R2	0436	
					13	13	00197	BEQL	15\$		
				44	AE	9F	00199	PUSHAB	MODNAME	0439	
					52	DD	0019C	PUSHL	R2		
	00000000G		00		02	FB	0019E	CALLS	#2, DBG\$STA_SYMNAME		
14	AE			28	A2	00	EF	001A5	EXTZV	#0, #1, 40(R2), MOD_SET_FLAG	0440
					14	AE	DD	001AC	PUSHL	MOD_SET_FLAG	0443
					53	DD	001AF	PUSHL	CURRENT_PC		
					7E	7C	001B1	CLRG	-(SP)		
					7E	7C	001B3	CLRG	-(SP)		
					016B	31	001B5	BRW	30\$		
			52		28	AE	DD	001B8	MOVL	SYM_RSTPTR, R2	0448
					15	12	001BC	BNEQ	17\$		
	00000000'				EF	9F	001BE	PUSHAB	P.AAF		
					01	DD	001C4	PUSHL	#1		
	00028362				8F	DD	001C6	PUSHL	#164706		
	00000000G		00		03	FB	001CC	CALLS	#3, LIB\$SIGNAL		
			54		0C	A2	DD	001D3	MOVL	12(R2), SYM_DSTPTR	0449
			4A		15	A2	E9	001D7	BLBC	21(R2), 19\$	0450
			5B			01	DD	001DB	MOVL	#1, CALL_FLAG	0462
					53	DD	001DE	PUSHL	CURRENT_PC	0463	
	0000V		CF		01	FB	001E0	CALLS	#1, FIND_MODRST		
	2C		AE		50	DD	001E5	MOVL	R0, MOD_RSTPTR		
				40	AE	9F	001E9	PUSHAB	SYMNAME	0464	
					52	DD	001EC	PUSHL	R2		
	00000000G		00		02	FB	001EE	CALLS	#2, DBG\$STA_SYMNAME		
			52		2C	AE	DD	001F5	MOD_RSTPTR, R2	0465	
					1E	13	001F9	BEQL	18\$		
				44	AE	9F	001FB	PUSHAB	MODNAME	0468	
					52	DD	001FE	PUSHL	R2		
	00000000G		00		02	FB	00200	CALLS	#2, DBG\$STA_SYMNAME		
			01		00	EF	00207	EXTZV	#0, #1, 40(R2), -(SP)	0471	
					53	DD	0020D	PUSHL	CURRENT_PC		
			7E		03	A4	C3	0020F	SUBL3	3(SYM_DSTPTR), CURRENT_PC, -(SP)	0470
					7E	7C	00214	CLRG	-(SP)	0469	
					0107	31	00216	BRW	29\$		
					53	DD	00219	PUSHL	CURRENT_PC	0477	
				7E	03	A4	C3	0021B	SUBL3	3(SYM_DSTPTR), CURRENT_PC, -(SP)	0476
					7E	7C	00220	CLRG	-(SP)	0475	
					0112	31	00222	BRW	32\$		
			06		14	A2	91	00223	CHPB	20(R2), #6	0482
					09	12	00229	BNEQ	20\$		
					53	DD	0022B	PUSHL	CURRENT_PC	0484	
					7E	7C	0022D	CLRG	-(SP)		
					7E	7C	0022F	CLRG	-(SP)		
					0106	31	00231	BRW	33\$		

[illegible]

				53	DD	002E1	28:	PUSHL	CURRENT_PC	0544
				01	FB	002E3		CALLS	#1, FIND_MODRST	
				50	DD	002E8		MOVL	R0, MOD_RSTPTR	
			40	AE	9F	002EC		PUSHAB	SYMNAME	0545
			2C	AE	DD	002EF		PUSHL	SYM_RSTPTR	
				02	FB	002F2		CALLS	#2, DBG\$STA_SYMNAME	
			18	AE	C3	002F9		SUBL3	STARTING_PC, CURRENT_PC, R5	0552
			2C	AE	DD	002FE		MOVL	MOD_RSTPTR, R2	0546
				29	13	00302		BEQL	318	
			44	AE	9F	00304		PUSHAB	MODNAME	0549
				52	DD	00307		PUSHL	R2	
				02	FB	00309		CALLS	#2, DBG\$STA_SYMNAME	
				00	EF	00310		EXTZV	#0, #1, 40(R2), -(SP)	0553
				53	DD	00316		PUSHL	CURRENT_PC	
				55	DD	00318		PUSHL	R5	0552
			44	AE	DD	0031A		PUSHL	STMT_NUMBER	0551
			4C	AE	DD	0031D		PUSHL	LINE_NUMBER	
			54	AE	DD	00320	29:	PUSHL	SYMNAME	0550
			5C	AE	DD	00323	30:	PUSHL	MODNAME	
				07	FB	00326		CALLS	#7, OUT_TRACEBACK	
				14	11	0032B		BRB	348	0546
				53	DD	0032D	31:	PUSHL	CURRENT_PC	0560
				55	DD	0032F		PUSHL	R5	0559
			40	AE	DD	00331		PUSHL	STMT_NUMBER	0558
			48	AE	DD	00334		PUSHL	LINE_NUMBER	
			50	AE	DD	00337	32:	PUSHL	SYMNAME	0557
				7E	D4	0033A	33:	CLRL	-(SP)	
				06	FB	0033C		CALLS	#6, OUT_TRACEBACK	
				5B	E9	00341	34:	BLBC	CALL_FLAG, 358	0569
				5B	D4	00344		CLRL	CALL_FLAG	0572
				AE	DD	00346		MOVL	SYM_RSTPTR, R0	0573
			28	AO	91	0034A		CMPB	20(R0), #2	
			14	29	12	0034E		BNEQ	358	
				53	D1	00350		CMPL	CURRENT_PC, 24(R0)	0576
				23	1F	00354		BLSSU	358	
				53	D1	00356		CMPL	CURRENT_PC, 28(R0)	0577
				1D	1A	0035A		BGTRU	358	
				AO	DD	0035C		MOVL	12(R0), SYM_DSTPTR	0580
			0C	A4	95	00360		TSTB	2(SYM_DSTPTR)	0581
			02	14	18	00363		BGEQ	358	
				EF	9F	00365		PUSHAB	P.AAG	0584
				01	FB	0036B		CALLS	#1, DBG\$PRINT	
				00	FB	00372		CALLS	#0, DBG\$NEWLINE	0586
				01	DD	00379	35:	MOVL	#1, EXC_TYPE	0599
				58	DD	0037D		MOVL	NEXT_FP, CURRENT_FP	0600
			10	A8	DD	00380		MOVL	16(NEXT_FP), CURRENT_PC	0601
			0C	A8	DD	00384		MOVL	12(NEXT_FP), NEXT_FP	0602
				6E	F2	00388	36:	AOBLSS	(SP), DEPTH, 378	0331
				04		0038D		RET		0610
				31		0038E	37:	BRW	38	0331
				FCDD						

; Routine Size: 913 bytes, Routine Base: DBG\$CODE + 0000


```
481 0611 1 ROUTINE OUT_TRACEBACK(MOD_NAM, LAB_NAM, LINE_NUM, STMT_NUM,  
482 0612 1 REL_PC, ABS_PC): NOVALUE =  
483 0613 1  
484 0614 1 FUNCTION  
485 0615 1 This routine actually calls FAO and DEBUG's output routine to  
486 0616 1 format and output a line of traceback information.  
487 0617 1  
488 0618 1 INPUTS  
489 0619 1 MOD_NAM - Address of a Counted ASCII string containing the module name.  
490 0620 1 LAB_NAM - Address of a Counted ASCII string containing the routine name.  
491 0621 1 LINE_NUM - Line number matching the PC.  
492 0622 1 STMT_NUM - Statement number within the LINE_NUM line.  
493 0623 1 REL_PC - Relative PC value from beginning of the routine.  
494 0624 1 ABS_PC - The absolute PC value from the call frame.  
495 0625 1  
496 0626 1 OUTPUTS  
497 0627 1 NONE  
498 0628 1  
499 0629 1  
500 0630 1  
501 0631 1 BEGIN  
502 0632 1  
503 0633 1 MAP  
504 0634 1 MOD_NAM: CS_POINTER, !  
505 0635 1 LAB_NAM: CS_POINTER; !  
506 0636 1  
507 0637 1  
508 0638 1 BUILTIN  
509 0639 1 ACTUALCOUNT, ! The number of actual parameters  
510 0640 1 ACTUALPARAMETER; ! Selects the N-th actual parameter  
511 0641 1  
512 0642 1  
513 0643 1 LOCAL  
514 0644 1 STRING_PTR: CS_POINTER; !  
515 0645 1  
516 0646 1  
517 0647 1 BIND  
518 0648 1 NULL_STRING = UPLIT BYTE (0);  
519 0649 1  
520 0650 1  
521 0651 1  
522 0652 1  
523 0653 1 ! Mark the module if the module is set.  
524 0654 1  
525 0655 1 IF ACTUALCOUNT() GTR 6  
526 0656 1 THEN  
527 0657 1 BEGIN  
528 0658 1 IF ACTUALPARAMETER(7)  
529 0659 1 THEN  
530 0660 1 DBG$PRINT(UPLIT BYTE(%ASCIC '*'))  
531 0661 1  
532 0662 1 ELSE  
533 0663 1 DBG$PRINT(UPLIT BYTE(%ASCIC ' '))  
534 0664 1  
535 0665 1 END  
536 0666 1  
537 0667 1 ELSE
```

```

538      0668      2      DBG$PRINT(UPLIT BYTE(%ASCIC ' '));
539      0669
540      0670
541      0671      ! Print the module name, if we have one.
542      0672
543      0673      STRING_PTR = .MOD_NAM;
544      0674      IF .MOD_NAM EQL 0 THEN STRING_PTR = NULL_STRING;
545      0675      DBG$PRINT(UPLIT(%ASCIC '!15AC-'), .STRING_PTR);
546      0676
547      0677
548      0678      ! Print the routine name, if we have one.
549      0679
550      0680      STRING_PTR = .LAB_NAM;
551      0681      IF .LAB_NAM EQL 0 THEN STRING_PTR = NULL_STRING;
552      0682      IF .STRING_PTR[0] GTRU 31
553      0683      THEN
554      0684          BEGIN
555      0685              DBG$PRINT(UPLIT(%ASCIC '!63AC'), .STRING_PTR);
556      0686              DBG$NEWLINE();
557      0687              DBG$PRINT(UPLIT(%ASCIC '!49* '));
558      0688              END
559      0689
560      0690      ELSE
561      0691          DBG$PRINT(UPLIT(%ASCIC '!32AC'), .STRING_PTR);
562      0692
563      0693
564      0694      ! Print the line number if one is available.
565      0695
566      0696      IF .LINE_NUM NEQ 0
567      0697      THEN
568      0698          DBG$PRINT(UPLIT(%ASCIC '!5UL'), .LINE_NUM)
569      0699
570      0700      ELSE
571      0701          DBG$PRINT(UPLIT(%ASCIC '!5* '));
572      0702
573      0703
574      0704      ! Print the statement number if applicable.
575      0705
576      0706      IF .STMT_NUM NEQ 0
577      0707      THEN
578      0708          DBG$PRINT(UPLIT(%ASCIC '!.!4ZL'), .STMT_NUM)
579      0709
580      0710      ELSE
581      0711          DBG$PRINT(UPLIT(%ASCIC '!5* '));
582      0712
583      0713
584      0714      ! Print the absolute PC and then output the print line. Then return.
585      0715
586      0716      DBG$PRINT(UPLIT(%ASCIC '!9XL!10XL'), .REL_PC, .ABS_PC);
587      0717      DBG$NEWLINE();
588      0718      RETURN;
589      0719
590      0720      END;

```

.PSECT DBG\$PLIT,NOWRT, SHR, PIC,0

00	00	4C	58	30	31	21	4C	58	39	21	09	00154	P.AAT:	.ASCII	<9>\!9XL!10XL\<0><0>
00	00	00	00	00	00	00	00	00	00	00	04	0014C	P.AAS:	.ASCII	<4>\!5* \<0><0><0>
00	00	00	00	00	00	00	00	00	00	00	05	00144	P.AAR:	.ASCII	<5>\!4ZL\<0><0>
00	00	00	00	00	00	00	00	00	00	00	04	0013C	P.AAQ:	.ASCII	<4>\!5* \<0><0><0>
00	00	00	00	00	00	00	00	00	00	00	04	00134	P.AAP:	.ASCII	<4>\!5UL\<0><0><0>
00	00	00	00	00	00	00	00	00	00	00	05	0012C	P.AAO:	.ASCII	<5>\!32AC\<0><0>
00	00	00	00	00	00	00	00	00	00	00	05	0011C	P.AAM:	.ASCII	<5>\!49* \<0><0>
00	00	00	00	00	00	00	00	00	00	00	05	00114	P.AAL:	.ASCII	<6>\!15AC \<0>
00	00	00	00	00	00	00	00	00	00	00	06	00112	P.AAK:	.ASCII	<1>\! \
00	00	00	00	00	00	00	00	00	00	00	01	00110	P.AAJ:	.ASCII	<1>\! \
00	00	00	00	00	00	00	00	00	00	00	01	0010E	P.AAI:	.ASCII	<1>\!*
00	00	00	00	00	00	00	00	00	00	00	01	0010D	P.AAH:	.BYTE	0

NULL_STRING= P.AAH

.PSECT DBG\$CODE,NOWRT, SHR, PIC,0

55	00000000G	00	9E	00002	OUT_TRACEBACK:	WORD	Save R2,R3,R4,R5	0611
54	00000000G	00	9E	00009	MOVAB	DBG\$NEWLINE, R5		
53	00000000'	EF	9E	00010	MOVAB	DBG\$PRINT, R4		
06		6C	91	00017	CMQB	(AP), #6		0655
		0E	1B	0001A	BLEQU	2\$		
05	1C	AC	E9	0001C	BLBC	28(AP), 1\$		0658
	01	A3	9F	00020	PUSHAB	P.AAI		0660
		08	11	00023	BRB	3\$		
	03	A3	9F	00025	PUSHAB	P.AAJ		0663
		03	11	00028	BRB	3\$		
	05	A3	9F	0002A	PUSHAB	P.AAK		0668
64		01	FB	0002D	CALLS	#1, DBG\$PRINT		
52	04	AC	D0	00030	MOVL	MOD_NAM, STRING_PTR		0673
		03	12	00034	BNEQ	4\$		0674
52		63	9E	00036	MOVAB	NULL_STRING, STRING_PTR		
		52	DD	00039	PUSHL	STRING_PTR		0675
	07	A3	9F	0003B	PUSHAB	P.AAL		
64		02	FB	0003E	CALLS	#2, DBG\$PRINT		
52	08	AC	D0	00041	MOVL	LAB_NAM, STRING_PTR		0680
		03	12	00045	BNEQ	5\$		0681
52		63	9E	00047	MOVAB	NULL_STRING, STRING_PTR		
1F		62	91	0004A	CMQB	(STRING_PTR), #31		0682
		13	1B	0004D	BLEQU	6\$		
		52	DD	0004F	PUSHL	STRING_PTR		0685
	0F	A3	9F	00051	PUSHAB	P.AAM		
64		02	FB	00054	CALLS	#2, DBG\$PRINT		
65		00	FB	00057	CALLS	#0, DBG\$NEWLINE		0686
	17	A3	9F	0005A	PUSHAB	P.AAN		0687
64		01	FB	0005D	CALLS	#1, DBG\$PRINT		
		08	11	00060	BRB	7\$		0682
		52	DD	00062	PUSHL	STRING_PTR		0691
	1F	A3	9F	00064	PUSHAB	P.AAO		
64		02	FB	00067	CALLS	#2, DBG\$PRINT		
	0C	AC	D3	0006A	TSTL	LINE_NUM		0696
		0B	13	0006D	BEQL	8\$		

DBGTBK
V04-000

E 10
16-Sep-1984 02:44:53
14-Sep-1984 12:17:53

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DEBUG.SRC]DBGTBK.B32;1
Page 19
(4)

	0C	AC	DD	0006F	PUSHL	LINE_NUM	: 0698
	27	A3	9F	00072	PUSHAB	P.AAP	
64		02	FB	00075	CALLS	#2, DBG\$PRINT	
		06	11	00078	BRB	9%	
	2F	A3	9F	0007A	8%:	PUSHAB	P.AAQ
64		01	FB	0007D	CALLS	#1, DBG\$PRINT	: 0701
	10	AC	D5	00080	9%:	TSTL	STMT_NUM
		0B	13	00083	BEQL	10%	: 0706
	10	AC	DD	00085	PUSHL	STMT_NUM	: 0708
	37	A3	9F	00088	PUSHAB	P.AAR	
64		02	FB	0008B	CALLS	#2, DBG\$PRINT	
		06	11	0008E	BRB	11%	
	3F	A3	9F	00090	10%:	PUSHAB	P.AAS
64		01	FB	00093	CALLS	#1, DBG\$PRINT	: 0711
7E	14	AC	7D	00096	11%:	MOVQ	REL_PC, -(SP)
	47	A3	9F	0009A	PUSHAB	P.AAT	: 0716
64		03	FB	0009D	CALLS	#3, DBG\$PRINT	
65		00	FB	000A0	CALLS	#0, DBG\$NEWLINE	: 0717
			04	000A3	RET		: 0720

; Routine Size: 164 bytes, Routine Base: DBG\$CODE + 0391

```

592 0721 1 ROUTINE FIND_MODRST(VALUE) =
593 0722 1
594 0723 1 FUNCTION
595 0724 1 This routine goes through the Program Static Address Table and
596 0725 1 test to see if the VALUE is within the module address range
597 0726 1 described in Program SAT. If VALUE is within the range then Module
598 0727 1 RST is returned.
599 0728 1
600 0729 1 INPUTS
601 0730 1 VALUE - The virtual address for which the corresponding module is
602 0731 1 to be found.
603 0732 1
604 0733 1 OUTPUTS
605 0734 1 Return value is Module RST pointer for the given symbol or 0
606 0735 1 if no Module RST pointer can be found.
607 0736 1
608 0737 1
609 0738 2 BEGIN
610 0739 2
611 0740 2 LOCAL
612 0741 2 SATPTR: REF SAT$ENTRY; ! Pointer to Program SAT entry
613 0742 2
614 0743 2
615 0744 2 ! Search through the Program Static Address Table until an entry is found
616 0745 2 ! (if any) which covers the specified address. If one is found, return a
617 0746 2 ! pointer to the corresponding Module RST Entry.
618 0747 2
619 0748 2 SATPTR = DBG$SEARCH_BIN_SAT (.SAT$START_ADDR, .VALUE, FALSE, TRUE);
620 0749 2 IF .SATPTR NEQ 0
621 0750 2 THEN
622 0751 2 RETURN .SATPTR[SAT$L_RSTPTR];
623 0752 2
624 0753 2 ! No module was found which contains the given virtual address.
625 0754 2 ! Return zero to indicate this.
626 0755 2
627 0756 2 RETURN 0;
628 0757 1 END;

```

0000 00000 FIND_MODRST:		WORD	Save nothing	0721
	01 DD 00002	PUSHL	#1	0748
	7E D4 00004	CLRL	-(SP)	
	04 AC DD 00006	PUSHL	VALUE	
00000000G	00 DD 00009	PUSHL	SAT\$START_ADDR	
	04 FB 0000F	CALLS	#4, DBG\$SEARCH_BIN_SAT	
	50 D5 00016	TSTL	SATPTR	0749
	05 13 00018	BEQL	1\$	
50	0C A0 D0 0001A	MOVL	12(SATPTR), R0	0751
	04 0001E	RET		
	50 D4 0001F 1\$:	CLRL	R0	0756
	04 00021	RET		0757

: Routine Size: 34 bytes. Routine Base: DBG\$CODE + 0435

: 629 0758 0 END ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
DBG\$PLIT	352	NOVEC,NOWRT, RD ; EXE, SHR, LCL, REL, CON, PIC,ALIGN(0)
DBG\$CODE	1111	NOVEC,NOWRT, RD ; EXE, SHR, LCL, REL, CON, PIC,ALIGN(0)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	8	0	1000	00:01.9
\$255\$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1	32	0	0	7	00:00.1
\$255\$DUA28:[DEBUG.OBJ]DBGLIB.L32;1	1545	78	5	97	00:01.9
\$255\$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1	418	103	24	31	00:00.4
\$255\$DUA28:[DEBUG.OBJ]DBGMSG.L32;1	386	7	1	22	00:00.3
\$255\$DUA28:[DEBUG.OBJ]DBGGEN.L32;1	150	2	1	12	00:00.3

: Information: 1
: Warnings: 0
: Errors: 0

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:DBGTBK/OBJ=OBJ\$:DBGTBK MSRC\$:DBGTBK/UPDATE=(ENH\$:DBGTBK)

: Size: 1111 code + 352 data bytes
: Run Time: 00:24.7
: Elapsed Time: 00:28.1
: Lines/CPU Min: 1838
: Lexemes/CPU-Min: 8839
: Memory Used: 273 pages
: Compilation Complete

0096 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY